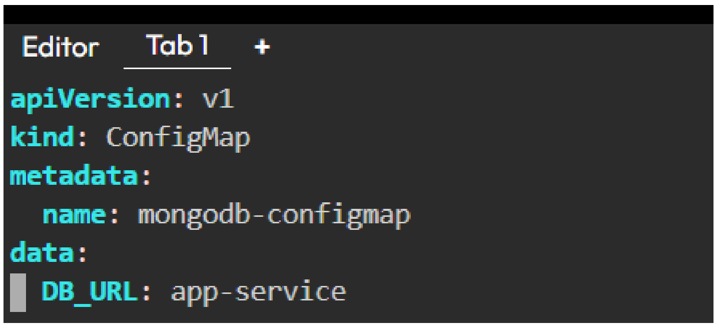
LAB 3

1) Create ConfgMap or MongoDB EndPoint. (The MondoDB sevice name)

DB_URL:mongo-service

name of clusterIP service attached to db-deployment

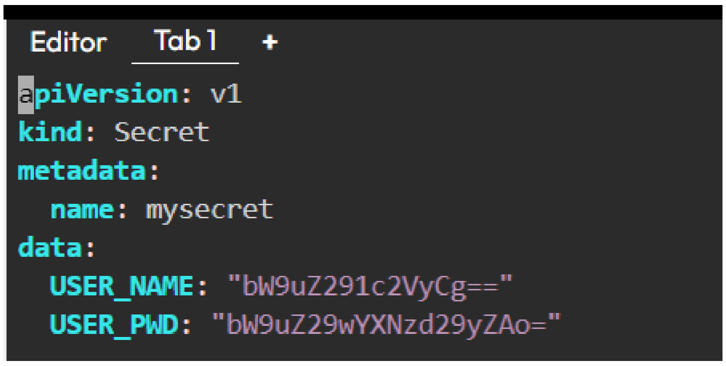


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-configmap
data:
  DB_URL: app-service
```

2) Create A secret or MongoDB User & PWD

USER_NAME: mongouser

USER_PWD: mongopassword



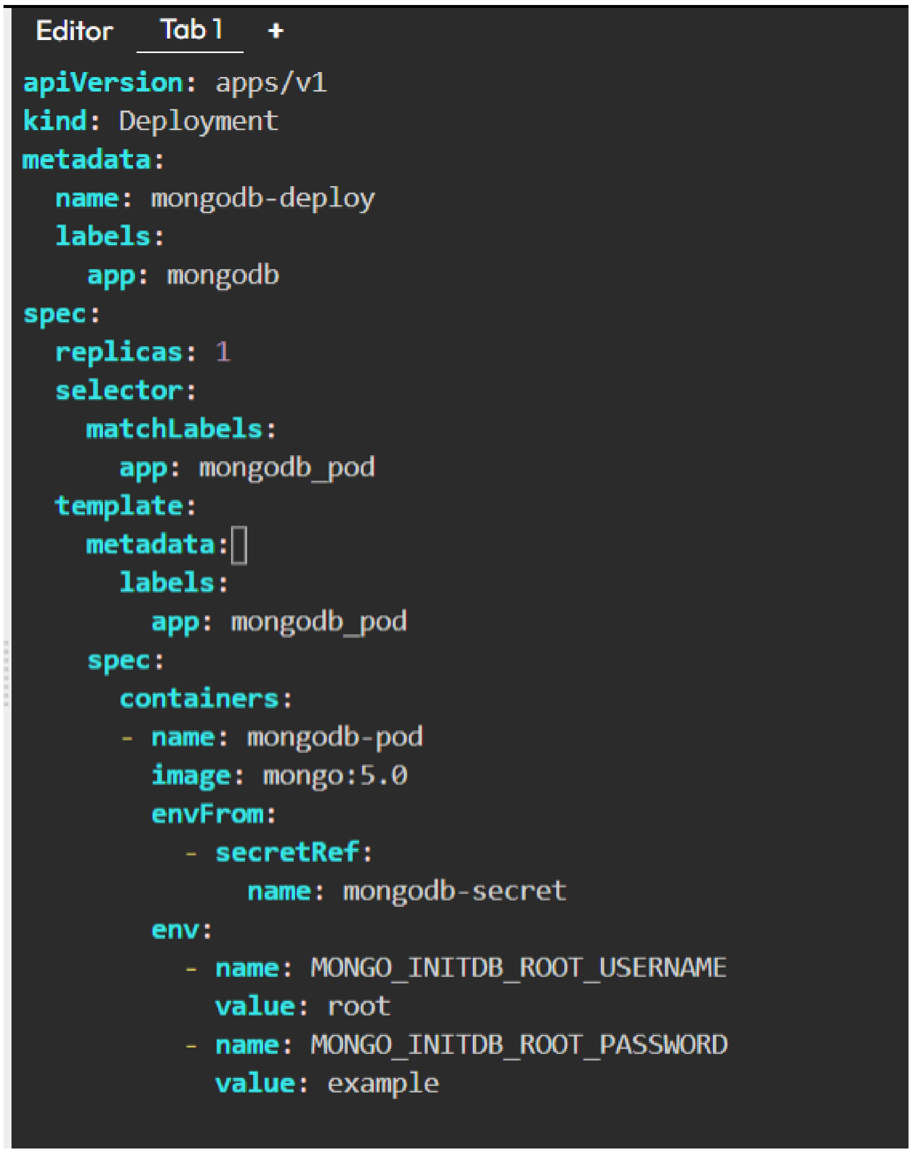
```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
data:
  USER_NAME: "bW9uZ291c2VyCg=="
  USER_PWD: "bW9uZ29wYXNzd29yZAo="
```

3) Create MongoDB Deployment Applicaton with Internal service (ClusterIp) Mongo DB needs username + password to operate

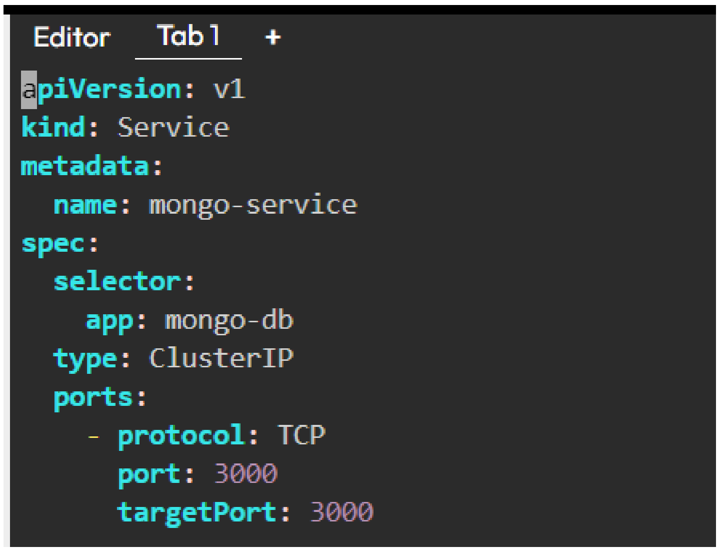
Vars needed in mongoDB:

MONGO_INITDB_ROOT_USERNAME: root

MONGO_INITDB_ROOT_PASSWORD: example

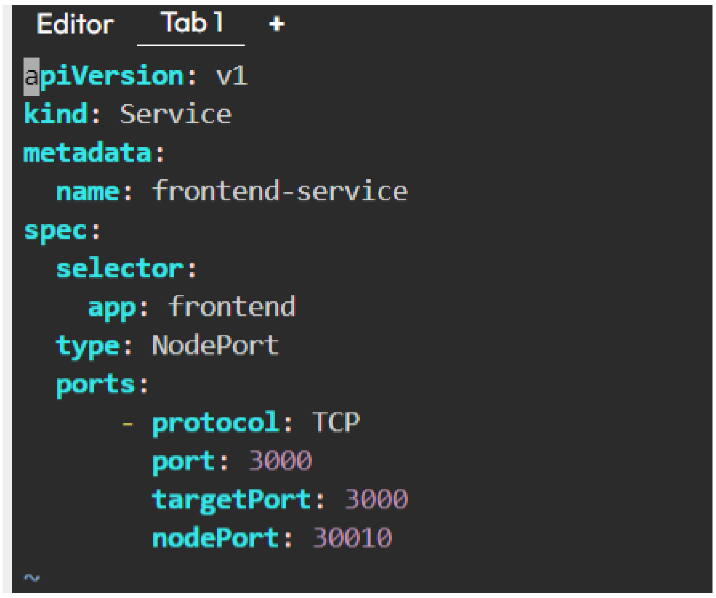


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deploy
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb_pod
  template:
    metadata:
      labels:
        app: mongodb_pod
    spec:
      containers:
      - name: mongodb-pod
        image: mongo:5.0
        envFrom:
          - secretRef:
              name: mongodb-secret
        env:
          - name: MONGO_INITDB_ROOT_USERNAME
            value: root
          - name: MONGO_INITDB_ROOT_PASSWORD
            value: example
```

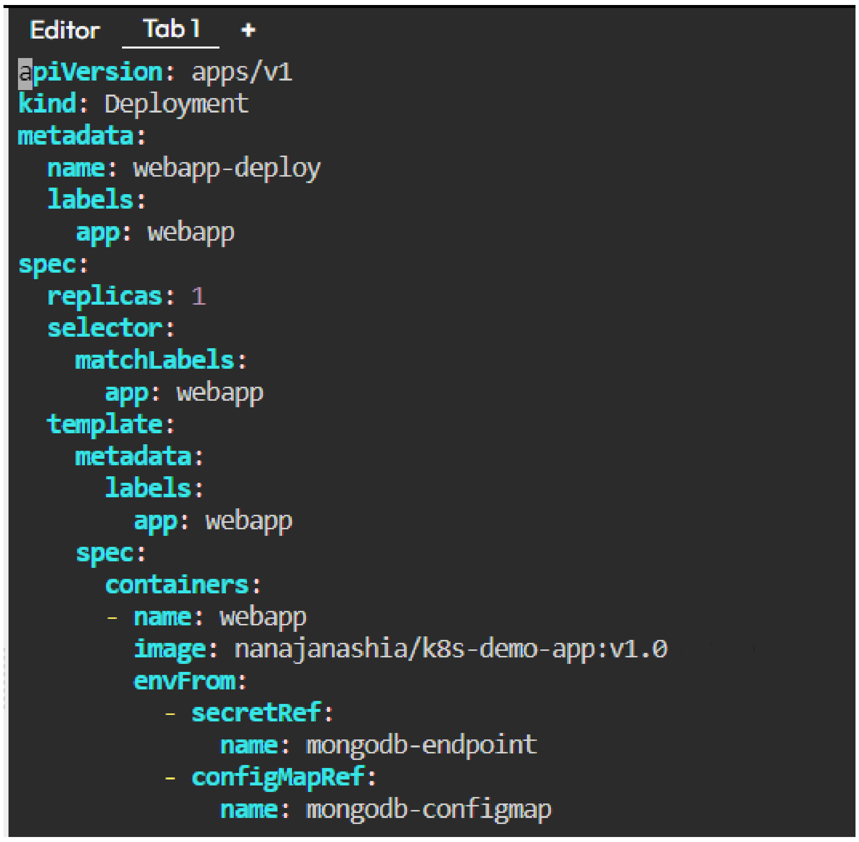


```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongo-service
spec:
  selector:
    app: mongo-db
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
```

4) Create webApp Deployment(FrontEnd(with external service) and it needs to access MongoDb, so it needs username+ password + mongodb endpoint (mongodb service) container runs on 3000

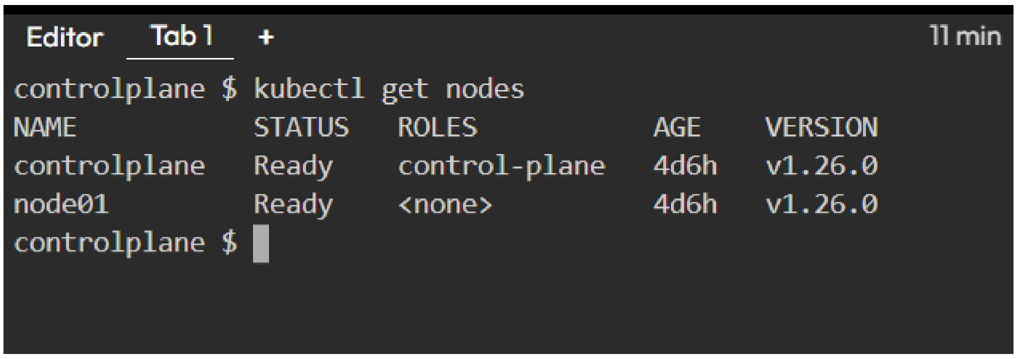


```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  type: NodePort
  ports:
      - protocol: TCP
        port: 3000
        targetPort: 3000
        nodePort: 30010
~
```



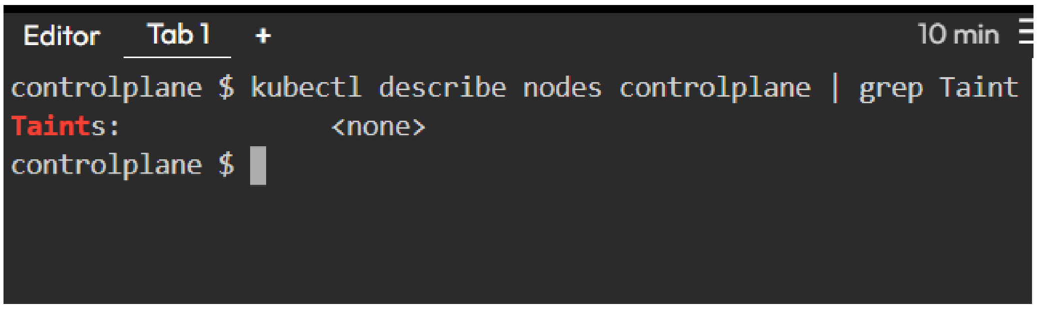
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp-deploy
  labels:
    app: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: nanajanashia/k8s-demo-app:v1.0
        envFrom:
          - secretRef:
              name: mongodb-endpoint
          - configMapRef:
              name: mongodb-configmap
```

8) How many Nodes exist on the system?



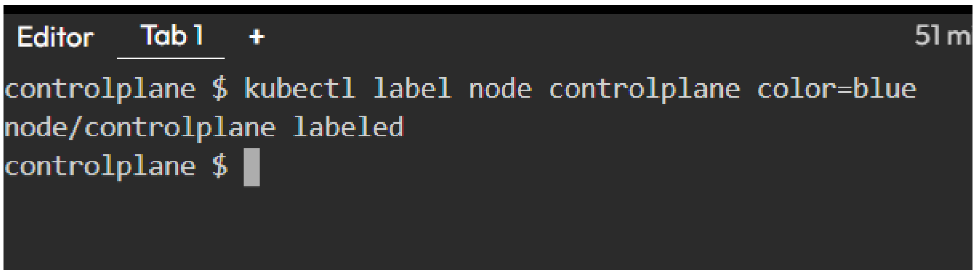
```
controlplane $ kubectl get nodes
NAME           STATUS   ROLES           AGE    VERSION
controlplane   Ready    control-plane   4d6h   v1.26.0
node01         Ready    <none>          4d6h   v1.26.0
controlplane $
```

9) Do you see any taints on master?



```
controlplane $ kubectl describe nodes controlplane | grep Taint
Taints:             <none>
controlplane $
```

10) Apply a label color=blue to the master node



```
controlplane $ kubectl label node controlplane color=blue
node/controlplane labeled
controlplane $
```

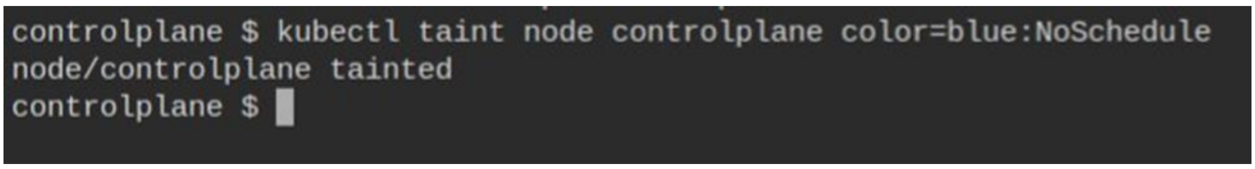
11) Create a new deployment named blue with the nginx image and 3 replicas

Set Node Afnity to the deployment to place the pods on master only

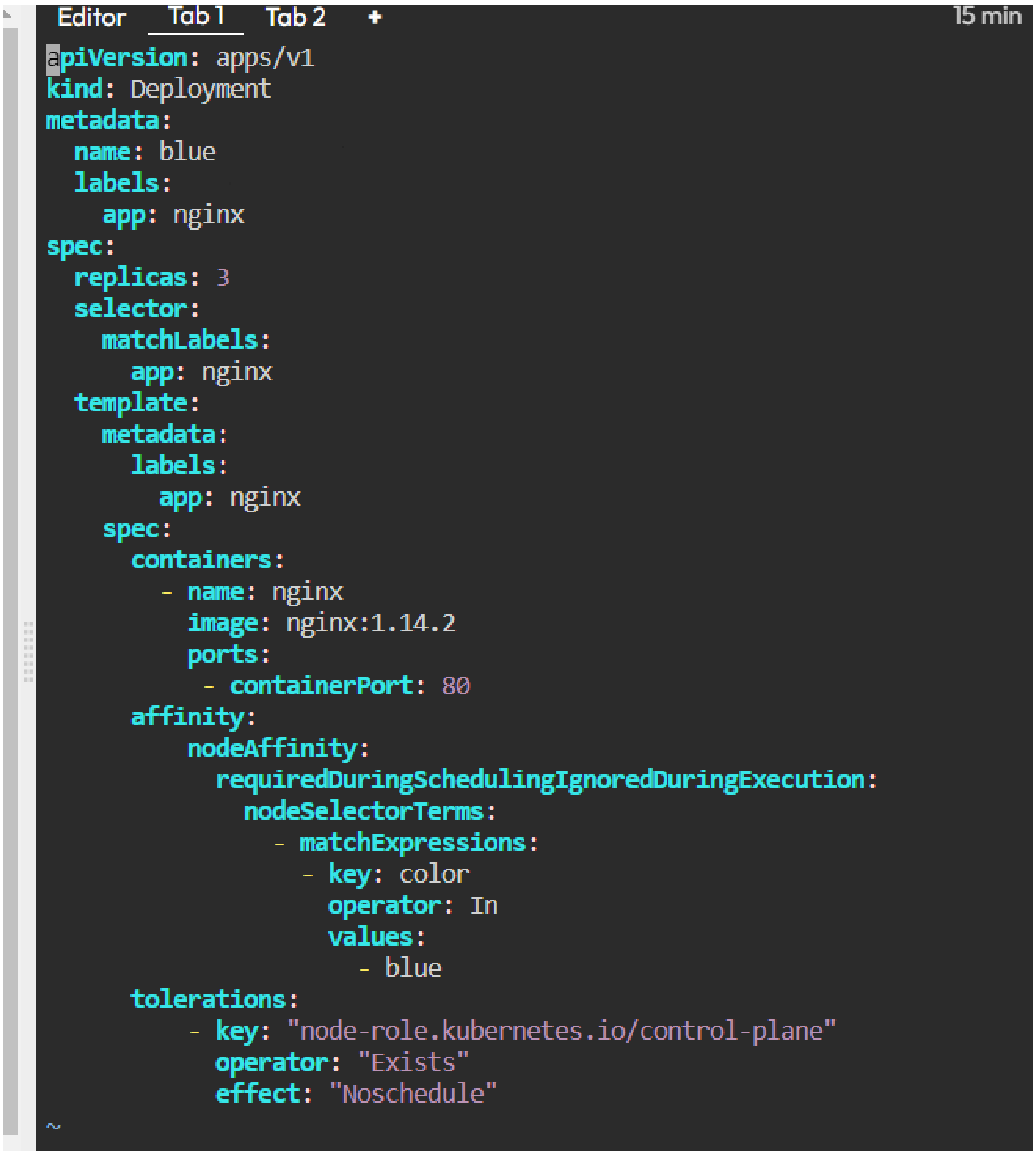
NodeAfnity: requiredDuringSchedulingIgnoredDuringExecuton

Key: color

values: blue

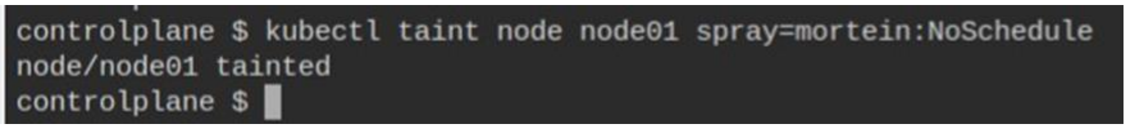


```
controlplane $ kubectl taint node controlplane color=blue:NoSchedule
node/controlplane tainted
controlplane $
```



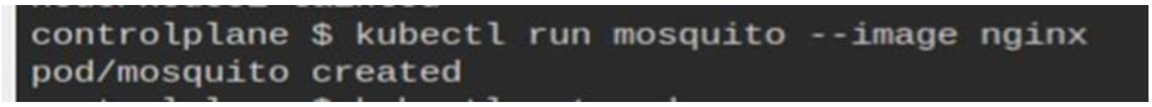
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blue
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
           - containerPort: 80
      affinity:
          nodeAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
              nodeSelectorTerms:
                - matchExpressions:
                  - key: color
                    operator: In
                    values:
                      - blue
      tolerations:
          - key: "node-role.kubernetes.io/control-plane"
            operator: "Exists"
            effect: "Noschedule"
~
```

12) Create a taint on node01 with key o spray, value o mortein and effect of NoSchedule



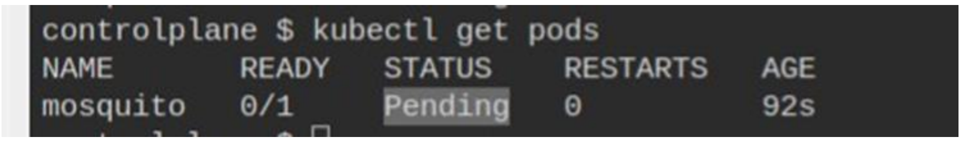
```
controlplane $ kubectl taint node node01 spray=mortein:NoSchedule
node/node01 tainted
controlplane $ 
```

13) Create a new pod with the NGINX image, and Pod name as mosquito



```
controlplane $ kubectl run mosquito --image nginx
pod/mosquito created
```

14) What is the state of the mosquito POD?



```
controlplane $ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          92s
```

15) Create another pod named bee with the NGINX image, which has a toleraton set to

the taint Mortein

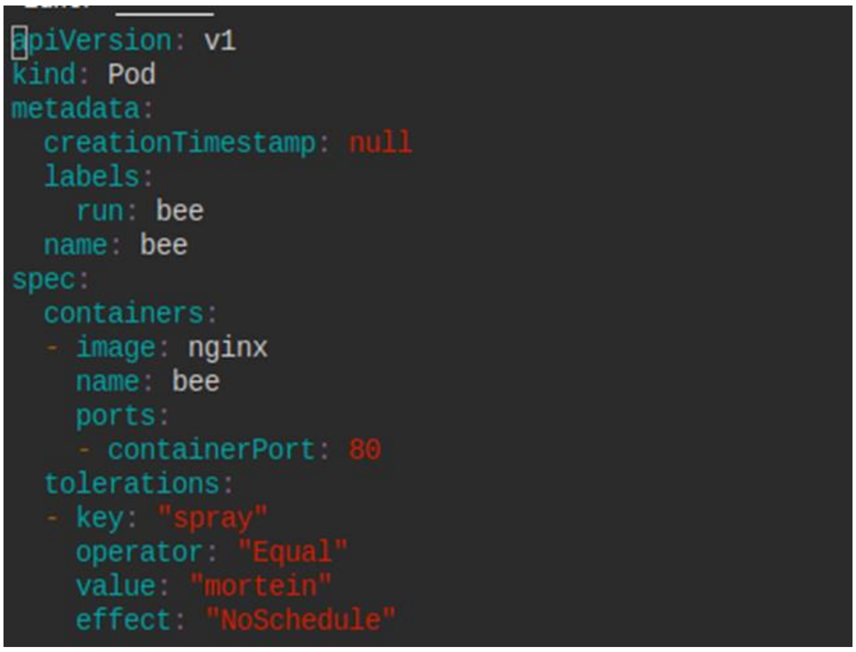
Image name: nginx

Key: spray

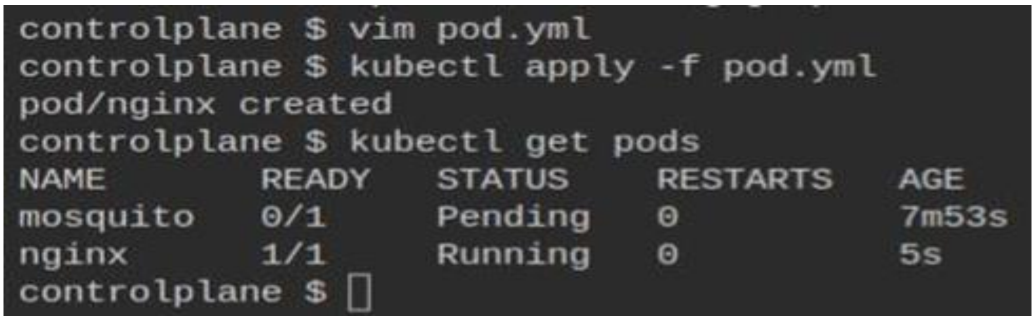
Value: mortein

Efect: NoSchedule

Status: Running



```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: bee
  name: bee
spec:
  containers:
  - image: nginx
    name: bee
    ports:
    - containerPort: 80
  tolerations:
  - key: "spray"
    operator: "Equal"
    value: "mortein"
    effect: "NoSchedule"
```



```
controlplane $ vim pod.yml
controlplane $ kubectl apply -f pod.yml
pod/nginx created
controlplane $ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          7m53s
nginx      1/1     Running   0          5s
controlplane $ 
```